Agence de location de Voitures

I. Etude de cas :

Nous souhaitons réaliser une solution de gestion d'une agence de location de Voitures.

Les classes principales de ce programme sont : la classe Voiture, la classe Client et la classe Agence.

Une voiture est caractérisée par son immatriculation, sa marque et son prix. Deux voitures sont égales s'ils ont la même immatriculation et la même marque.

Cette agence offre à ces clients la possibilité de choisir les voitures à louer en fonction de différents critères tels que la marque et le prix aussi. Il est possible de sélectionner dans la liste des voitures à louer toutes les voitures satisfaisant un critère donné.

Une agence est caractérisée par un nom, un parking qui permet d'enregistrer l'ensemble des voitures de l'agence de location ainsi qu'elle souhaite sauvegarder l'ensemble de ces clients.

Les fonctions standard pour la gestion de location de voiture sont :

- ✓ retourner la liste des voitures selon un critère défini
- ✓ afficher les clients et leurs voitures louées.
- ✓ louer une voiture à un client.
- ✓ retourner une voiture louée par un client,
- ✓ retourner l'ensemble des clients qui ont loué une (des) voiture(s)
- ✓ retourner la liste des voitures actuellement en état de location

Travail demandé :

Q1. Implémenter la classe Voiture.

Q2. Implémenter la classe Client sachant qu'un client est caractérisé par un code, un nom et un prénom. Deux clients sont égaux s'ils ont le même code.

II. Les Interfaces

Afin de choisir les critères des voitures, on définit l'interface Critère ainsi :

```
public interface Critere {
boolean estSatisfaitPar(Voiture v);
}
```

Q3. Implémenter la classe CritereMarque (Choix selon la marque)

Q4. Implémenter la classe CriterePrix (Choix selon le prix)

Q5. Classe Main.java :

1) Un client demande de louer une voiture de marque « Renault » et un prix de 300dt/j
2) Tester parmi les voitures disponibles, est ce que l'agence peut satisfait le client

Voiture	Marque	Prix de location
V1	Renault	250/j
V2	Fiat	300/j
V3	Renault	300dt/j
V24	Renault	500/j

III. Collection : LIST <Vector & ArrayList>

Cette agence a mis à la disposition de ces clients la possibilité de louer plusieurs voitures, pour cela on se propose de créer une nouvelle classe **ListVoitures** qui permet de stocker un ensemble de voiture dans une liste.

Cette classe permet d'ajouter, de supprimer une voiture et d'afficher la liste des voitures.

Q6. Classe ListVoitures compléter les méthodes suivants :

1) public void addVoiture(Voiture v)
2) public void removeVoiture(Voiture v)
3) public void existe (Voiture v)

NB : pour toutes informations, voir dans le support du cours. Plus précisément, nous allons utiliser soit un « Vector » ou un « ArrayList ».

Q7. Classe Main.java :

4) Tester les différentes méthodes d'ajout et de suppression de la classe ListeVoitures.
5) Afficher la liste de voitures.
6) Afficher la liste de voiture triée selon leurs immatriculations.
7) Afficher aussi la liste de voiture triée selon leurs marques.
8) Afficher la liste de voitures (utiliser l'interface **Iterator** pour parcourir la liste des voitures)

IV. Collection :SET <HashSet& TreeSet>

On se propose de créer l'ensemble de ces clients afin de sauvegarder les clients de l'agence,

Q8. Classe **SetClient** compléter les méthodes suivants :

1) Créer un Set de clients
2) public void ajouterClient(Client c)

NB : Sachant qu'on s'est demandé de ne pas accepter d'ajouter deux clients identiques

3) public void supprimerClient(Client c)
4) public void trouverClient(Client c)
5) public boolean chercherClient(Client c)
6) public void afficherClient(Client c), qu'est la méthode adéquate pour parcourir le Set ?
7) Remplacer le Set déjà utilisé par un Set qui permet d'avoir des clients ordonnés selon leurs identifiants (faire les modifications nécessaires)